UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P O Box 1450
Alexandria, Virginia 22313-1450
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/656,097	09/05/2003	Bhushan Khaladkar	OI7035732001	9920

55498 7590 12/28/2009
ORACLE INTERNATIONAL CORPORATION
c/o VISTA IP LAW GROUP LLP
1885 LUNDY AVENUE
SUITE 108
San Jose, CA 95131

EXAMINER	
TSUI, WILSON W	

ART UNIT	PAPER NUMBER
2178	

MAIL DATE	DELIVERY MODE
12/28/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)
	10/656,097	KHALADKAR ET AL.
	Examiner	Art Unit
	WILSON TSUI	2178

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 August 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-62* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-62* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

DETAILED ACTION

1. This final action is in response to the amendment filed on: 08/28/09.

2. Claims 1, 22, 18, 30, 31, 32, 33, 34, and 35 are amended. Claims 1-62 are pending.

3. The following rejections are withdrawn, in view of new grounds of rejection necessitated by applicant's amendments:

- Claims 1-4, 8, 10, 11, 12, 14, 18 – 21, 23, 24, 29, 30 – 38, 40, 42, 43, 45, 47, 48, 50 - 52, 54, 56, 60 rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy, in view of Dreyband et al, and further in view of Bhatt et al.
- Claims 15-17, 25-28, 57, 58, 61, and 62 rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy, in view of Dreyband et al, in view of Bhatt et al, and further in view of Wan.
- Claims 5, 6, 7, 9, 13, 22, 39, 41, 44, 46, 49, 53, 55, and 59 rejected under 35 U.S.C. 103(a) as being unpatentable over Marcy, in view of Dreyband et al, in view of Bhatt et al, and further in view of JAXB.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1, 2, 4, 8, 10, 18, 19, 21, 23, 24, 29, 30-38, 40, 42, 43, 45, 47, 48, 50, 51, 52, 54, 56, 60 are rejected under 35 U.S.C. 103(a) as being unpatentable over Warshavsky et al (US Patent: 6,732,095 B1, issued: May 4, 2004, filed: Apr. 13, 2001) in view of Dreyband et al (US Patent Application: US 2001/0029604 A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), and further in view of Wang et al ("Discovering Typical Structures of Documents: A Road Map Approach", publisher: ACM, published: 1998).

With regards to claim 1,

Warshavsky et al teaches *receiving a schema for the XML data* (column 3, lines 58-62: whereas a DTD or schema is received)*;*

Storing the XML data in a database in a storage system, wherein one or more elements of the schema are stored in one or more respective columns of the database (column 3, lines 45-56, column 4, lines 12-25: whereas, XML data is stored in a relational database having records and columns).

Identifying an element within the schema to associate with an access procedure (column 4, lines 12-25: whereas, elements are stored into a database for later retrieval/access via a XML converter)*;*

Determining, by using a processor if the element is appropriate for association with the access procedure based at least in part upon a datatype of the element (Table 1: whereas, valid string datatypes are stored, while undefined string datatypes are ignored)*, wherein certain datatypes of elements within the database are designated as not eligible for the access procedure such that the element having at least one of the*

certain datatypes are not associated with the access procedure (Table 1: whereas, undefined string datatypes are ignored and not accessed through an access procedure)*;*

If the element is appropriate for association, then creating the access procedure and associating the access procedure with the element, the access procedure providing direct access to an instance of the XML data such that the access procedure returns an appropriate data type for the element without converting the data type of the element (column 4, lines 12-20, Table 3: whereas, the appropriate string data type element values are returned/accessed using a XML converter), *wherein direct access comprises accessing a column of the database for the element to access an instance of the XML data associated with the element* (column 3, lines 50-56, column 4, lines 13-17: whereas, a column of a database is referenced)*; and*

Storing the element in a volatile or non-volatile computer readable medium or displaying the element on a display device (column 3, lines 14-25: whereas different types of volatile, and non-volatile mediums can be implemented).

However, the Warshavsky et al does not expressly teach the access procedure is a named access procedure, and *without progressive traversal of a hierarchy of elements defined in the schema.*

Yet, Dreyband et al teaches *a named access procedure* (Figure 3, paragraph 0029: whereas, elements defined in the xml file are identified, such as the element used in Dreyband's example xml file labeled: "name");.

It would have been obvious to one of the ordinary skill in the art a the time of the invention to have modified Warshavsky et al's xml data access procedure, which is based on a schema, such that the accessing of node data can be accomplished through named access, as taught by Dreyband et al. The combination of Warshavsky et al and Dreyband et al would have allowed Warshavksy et al to have implemented "the ability to effectively map a semantically descriptive language, such as Schema, into an object oriented language, without losing the relational characteristics of complex data structures (Dreyband et al, paragraph 0008).

However, although Warshavsky et al supports compressing tables of mapping data (column 3, lines 64-67), the combination of Warshavsky et al and Dreyband et al do not expressly teach *without progressive traversal of a hierarchy of elements defined in the schema*.

Yet, Wang teaches *without progressive traversal of a hierarchy of elements defined in the schema* (page 149, section 3, left column: whereas, the mapping of a hierarchy is compressed, such that frequently accessed nodes are mapped, rather than mapping all of the hierarchy of elements).

It would have been obvious to have modified Warshavsky et al and Dreyband et al's method for named access procedure, such that the named access is made possible without having to progressively traverse all the nodes in a hierarchy, as taught by Wang. The combination would have allowed Warshavsky et al to have "reduced the disk access" and access frequently accessed data more efficiently without linear traversal of

the entire schema (Wang, page 149, section 3, column 1, and page 154, section 6, left column).

With regards to claim 2, which depends on claim 1, for a method performing a similar method to claim 12, is rejected under the same rationale.

With regards to claim 4, which depends on claim 1, Warshavsky and Dreyband teaches a named access procedure (as similarly explained in the rejection for claim 1), and Dreyband et al further teaches a method *in which a named access procedure is defined to get a value for the child node or to set a value for the child node* (Figure 2 and 3: whereas, child elements disclosed in a schema shown in Figure 2 (such as child node 'Name'), have corresponding get and set functions shown in Figure 3 (such as 'getName' and 'setName')).

With regards to claim 8, which is dependent on claim 1, the combination of Warshavsky, Dreyband et al, and Wang teach *in which the element is not appropriate for association if it corresponds to a datatype that is not defined in the schema,* as similarly explained in the rejection for claim 1, and is rejected under similar rationale.

With regards to claim 10, which is dependent on claim 1, Warshavsky et al, Dreyband et al, and Wang teach *the named access procedure,* as similarly explained in the rejection for claim 1, and is rejected under similar rationale. Additionally, Dreyband

et al further teaches the named access procedure *is implemented as a bean accessor type* (Figure 2 and 3: whereas, child elements disclosed in a schema shown in Figure 2 (such as child node 'Name'), have corresponding get and set functions shown in Figure 3 (such as 'setName')).

With regards to claim 18, Warshavsky et al teaches *a method for implementing efficient access to data that is based at least in part upon a markup language, in which the data associated with a schema, the data comprising a parent node and one or more child nodes, the method comprising:*

Receiving the schema for the data that is based on the mark-up language (column 3, lines 58-62: whereas a DTD or schema is received)*;*

Storing the data in a database in a storage system, wherein the one or more child nodes of the data is stored in one or more respective columns of the database (column 3, lines 45-56, column 4, lines 12-25: whereas, XML data is stored in a relational database having records and columns)*;*

Identifying a child node that is to be access within the data based at least in part upon a datatype of a child node, wherein certain datatypes of child nodes within the database is designated as not eligible for an access procedure (column 4, lines 12-25: whereas, elements are stored into a database for later retrieval/access via a XML converter. Additionally, as shown in Table 1: valid string datatypes are stored, while undefined string datatypes are ignored))*;*

Reviewing the schema to determine one or more access parameters relating to the child node (column 3, lines 55-67: whereas, data in the schema is used to establish a mapping between XML and relational data)*;*

Determining, by using a processor, the one or more access parameters for the child node relative to the parent node in accordance with the schema, wherein the at least on access parameter allows the access procedure to have direct access to an instance of the child node stored in a column of the database, wherein direct access comprises accessing the column of the database for the child node to access an instance of the data associated with the child node (column 3, lines 40-56: whereas, a mapping is determined using hierarchical data from a schema for direct database/relational access)*;*

Using the one or more access parameters to directly access the instance of the child node within the data such that the access procedure returns an appropriate datatype for the child node without converting the datatype of the child node (column 4, lines 12-20, Table 3: whereas, the appropriate string data type element values are returned/accessed using a XML converter (which uses mapping and related metadata))*; and*

Storing the one or more access parameters in a volatile or non-volatile computer readable medium or displaying the one or more access parameters on a display device (column 3, lines 39-56: whereas mapping data is stored in definitions).

However, Warshavsky et al does not expressly teach the access procedure is a *named* access procedure, and *without progressive traversal of a hierarchy of nodes defined in the schema*.

Yet, Dreyband et al teaches *a named access procedure* (Figure 3, paragraph 0029: whereas, elements defined in the xml file are identified, such as the element used in Dreyband's example xml file labeled: "name");.

It would have been obvious to one of the ordinary skill in the art a the time of the invention to have modified Warshavsky et al's xml data access procedure, such that the accessing of node data can be accomplished through named access, as taught by Dreyband et al. The combination of Warshavsky et al and Dreyband et al would have allowed Warshavksy et al to have implemented "the ability to effectively map a semantically descriptive language, such as Schema, into an object oriented language, without losing the relational characteristics of complex data structures (Dreyband et al, paragraph 0008).

However, although Warshavsky et al supports compressing tables of mapping data (column 3, lines 64-67), the combination of Warshavsky et al and Dreyband et al do not expressly teach *without progressive traversal of a hierarchy of elements defined in the schema*.

Yet, Wang teaches *without progressive traversal of a hierarchy of elements defined in the schema* (page 149, section 3, left column: whereas, the mapping of a hierarchy is compressed, such that frequently accessed nodes are mapped, rather than mapping all of the hierarchy of elements).

It would have been obvious to have modified Warshavsky et al and Dreyband et al's method for named access procedure, such that the named access is made possible without having to progressively traverse all the nodes in a hierarchy, as taught by Wang. The combination would have allowed Warshavsky et al to have "reduced the disk access" and access frequently accessed data more efficiently without linear traversal of the entire schema (page 149, section 3, column 1, and page 154, section 6, left column).

With regards to claim 19, Warshavsky et al teaches a method *in which the mark-up language is based on XML,* as similarly explained in the rejection for claim 1, and is rejected under similar rationale.

With regards to claim 21, which depends on claim 18, the combination of Warshavsky et al, Dreyband et al, and Wang teaches *the named access procedure* (as similarly explained in the rejection for claim 1); and Dreyband et al further teaches the named access procedure *is defined to get a value for the child node or to set a value for the child node* (Figure 2 and 3: whereas, child elements disclosed in a schema shown in Figure 2 (such as child node 'Name'), have corresponding get and set functions shown in Figure 3 (such as 'getName' and 'setName')).

With regards to claim 23, which depends on 18, the combination of Warshavsky et al, Dreyband et al, and Wang et al similarly teaches a method in which *the child node*

is not directly accessed if the node is not defined in the schema (since the node is ignored as taught by Warshavsky et al), and is rejected under similar rationale.

With regards to claim 24, which depends on claim 18, the combination of Warshavsky et al, Dreyband et al, and Wang et al teaches a method comprising *directly accessing a child node in which the method is performed* (as similarly explained in the rejection for claim 18, and is rejected under the same rationale). Furthermore, Dreyband et al teaches implementing the method using Java (paragraph 0011), and thus it is inherent that a method implemented in Java will *support the system's native data types* using the system's Java Virtual Machine.

With regards to claim 29, which depends on claim 18, the combination of Warshavsky et al, Dreyband et al, and Wang et al teaches a method *in which other child nodes not presently needed are not loaded into memory* (as similarly explained in Wang et al for the rejection of claim 1; since infrequently used elements are opted to not be loaded into memory), and is rejected under similar rationale..

With regards to claim 30, Warshavsky et al teaches a system for implementing efficient access to XML data comprising:

Means for receiving a schema for the XML data (column 3, lines 58-62: whereas a DTD or schema is received);

A storage system for storing the XML data in a database, wherein one or more elements of the schema are stored in one or more respective columns of the database (column 3, lines 45-56, column 4, lines 12-25: whereas, XML data is stored in a relational database having records and columns);

Means for identifying an element within the schema to associate with an access procedure (column 4, lines 12-25: whereas, elements are stored into a database for later retrieval/access via a XML converter)*;*

A processor for determining if the element is appropriate for association with the access procedure based at least in part upon a datatype of the element (Table 1: whereas, valid string datatypes are stored, while undefined string datatypes are ignored)*, wherein certain datatypes of elements within the database are designated as not eligible for the access procedure such that the element having at least one of the certain datatypes are not associated with the access procedure* (Table 1: whereas, undefined string datatypes are ignored and not accessed through an access procedure)*;*

Means for creating the access procedure and associating the named access procedure with the element if the element is appropriate for association, the access procedure providing direct access to the element within the XML data such that the access procedure returns an appropriate datatype for the element without converting the datatype of the element (column 4, lines 12-20, Table 3: whereas, the appropriate string data type element values are returned/accessed using a XML converter)*; wherein direct access comprises accessing the column of the database for the element to*

access an instance of the XML data associated with the element (column 3, lines 50-56, column 4, lines 13-17: whereas, a column of a database is referenced)*; and a volatile or non-volatile computer readable medium for storing the at least one parameter or a display device for displaying the at least one parameter* (column 3, lines 14-25: whereas different types of volatile, and non-volatile mediums can be implemented).

However, Warshavsky et al does not expressly teach the access procedure is a *named* access procedure, and *without progressive traversal of a hierarchy of elements defined in the schema.*

Yet, Dreyband et al teaches *a named access procedure* (Figure 3, paragraph 0029: whereas, elements defined in the xml file are identified, such as the element used in Dreyband's example xml file labeled: "name")*;*

It would have been obvious to one of the ordinary skill in the art a the time of the invention to have modified Warshavsky et al's xml data access procedure, such that the accessing of node data can be accomplished through named access, as taught by Dreyband et al. The combination of Warshavsky et al and Dreyband et al would have allowed Warshavksy et al to have implemented "the ability to effectively map a semantically descriptive language, such as Schema, into an object oriented language, without losing the relational characteristics of complex data structures (Dreyband et al, paragraph 0008).

However, although Warshavsky et al supports compressing tables of mapping data (column 3, lines 64-67), the combination of Warshavsky et al and Dreyband et al do not

expressly teach *without progressive traversal of a hierarchy of elements defined in the schema*.

Yet, Wang teaches *without progressive traversal of a hierarchy of elements defined in the schema* (page 149, section 3, left column: whereas, the mapping of a hierarchy is compressed, such that frequently accessed nodes are mapped, rather than mapping all of the hierarchy of elements).

It would have been obvious to have modified Warshavsky et al and Dreyband et al's method for named access procedure, such that the named access is made possible without having to progressively traverse all the nodes in a hierarchy, as taught by Wang. The combination would have allowed Warshavsky et al to have “reduced the disk access” and access frequently accessed data more efficiently without linear traversal of the entire schema (Wang, page 149, section 3, column 1, and page 154, section 6, left column).

With regards to claim 31 for a computer program product comprising a computer usable medium having executable code, is similar to a system performing a similar method to claim 30, and is rejected under the same rationale.

With regards to claim 32, for a system performing a method similar to claim 11, and is rejected under the same rationale.

With regards to claim 33, for a computer program product comprising a computer usable medium having executable code, is similar to a system performing a method of claim 11, and is rejected under the same rationale.

With regards to claim 34, for a system performing a method similar to claim 18, and is rejected under the same rationale.

With regards to claim 35, for a computer program product comprising a computer usable medium having executable code, performing a method similar to claim 18, and is rejected under the same rationale.

With regards to claim 36, which depends on claim 18, Warshavsky et al teaches *direct access is performed to a location in an XML document for the child node,* as similarly explained in the rejection for claim 18, and is rejected under similar rationale.

With regards to claim 37, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 1, is rejected under the same rationale.

With regards to claim 38, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 3, is rejected under the same rationale.

With regards to claim 40, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 8, is rejected under the same rationale.

With regards to claim 42, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 2, is rejected under the same rationale.

With regards to claim 43, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 3, is rejected under the same rationale.

With regards to claim 45, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 8, is rejected under the same rationale.

With regards to claim 47, which depends on claim 34, for a system performing a method that is similar to the method of claim 18, is rejected under the same rationale.

With regards to claim 48, which depends on claim 34, for a system performing a method that is similar to the method of claim 20, is rejected under the same rationale.

With regards to claim 50, which depends on claim 34, for a system performing a method that is similar to the method of claim 24, is rejected under the same rationale.

With regards to claim 51, which depends on claim 35, for a computer program product performing a method that is similar to the method of claim 18, is rejected under the same rationale.

With regards to claim 52, which depends on claim 35, for a computer program product performing a method that is similar to the method of claim 20, is rejected under the same rationale.

With regards to claim 54, which depends on claim 35, for a computer program product performing a method that is similar to the method of claim 24, is rejected under the same rationale.

With regards to claim 56, which depends on claim 32, for a system performing a method similar to the method of claim 14, is rejected under the same rationale.

With regards to claim 60, which depends on claim 33, for a computer program product performing a method that is similar to the method of claim 14, is rejected under the same rationale.

5. Claims 3 and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over Warshavsky et al (US Patent: 6,732,095 B1, issued: May 4, 2004, filed: Apr. 13, 2001) in view of Dreyband et al (US Patent Application: US 2001/0029604 A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), in view of Wang ("Discovering Typical Structures of Documents: A Road Map Approach", publisher: ACM, published: 1998), and further in view of Marcy (US Patent: 6,662,342 B1, published: Dec. 9, 2003, filed: Dec. 13, 1999).

With regards to claim 3, which depends on claim 2, Warshavsky et al does not teach teaches a method comprising *access parameters includes offset information for each element*.

Yet, Marcy teaches *access parameters includes offset information for each element* (whereas each element (for which the elements include child nodes, as shown in Figure 5) in a XML document (Figure 1) is assigned a memory offset location to be sent to an application for use as access parameters (column 6, lines 60-65)).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky et al's method for accessing/retrieving XML data, such that offset information is implemented, as taught by Marcy. The combination would have allowed Warshavsky et al to have "implemented an improved method for scanning XML data" (Marcy, column 2, lines 45-47).

With regards to claim 20, which depends on claim 19, the combination of Warshavsky et al, Dreyband et al, Wang, and Marcy similarly teaches a method *in which at least one access parameters is based on offset position,* as similarly explained in the rejection for claim 3, and is rejected under similar rationale.

6. Claims 25-28, 57, 58, 61 and 62 are rejected under 35 U.S.C. 103(a) as being unpatentable over Warshavsky et al (US Patent: 6,732,095 B1, issued: May 4, 2004, filed: Apr. 13, 2001) in view of Dreyband et al (US Patent Application: US 2001/0029604 A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), in view of Wang ("Discovering Typical Structures of Documents: A Road Map Approach", publisher: ACM, published: 1998), and further in view of Wan (US Patent: 2003/0233618 A1, published: Dec. 18, 2003, filed: Jun. 16, 2003, Foreign priority: Jun. 17, 2002).

With regards to claim 25, which depends on claim 18, the combination of Warshavsky et al, Dreyband et al, and Wang et al do not expressly teach a method *in which direct access is performed to an offset location for the child node.*

However, Wan teaches a method *in which direct access is performed to an offset location for the child node* (Table B and D: whereas, Table B represents an XML instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes using offsets).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky et al's application for accessing node data, to

further include the method for accessing child nodes via offsets as taught by Wan. The combination of Warshavsky et al, Dreyband et al, Wang, and Wan would have allowed system to have parsed “schemas of structural documents to determine predefined deterministic relationships between components of structured documents to be indexed” (paragraph 0009).

With regards to claim 26, which depends on claim 25, the combination of Warshavsky et al, Dreyband et al, and Wang et al do not expressly teach a method *in which the child node is at a known offset from a location for the parent node*.

However, Wan teaches a method *in which the child node is at a known offset from a location for the parent node* (Table B and D: whereas, Table B represents an XML instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes using offsets based off of the memory location of a parent node).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky et al's application for accessing node data to further include the method for accessing child nodes via offsets based off of a location of the corresponding parent nodes as taught by Wan. The combination of Warshavsky et al, Dreyband et al, Wang et al, and Wan would have allowed Warshavsky et al's system to have been able to have reduced iterative traversal of an XML instance.

With regards to claim 27, which depends on claim 26, the combination of Warshavsky et al, Dreyband et al, Wang, and Wan similarly *teach in which a mapping of the known offset is managed independently of the data* (as shown in table D of Wan), and is rejected under similar rationale.

With regards to claim 28, which depends on claim 25, the combination of Warshavsky et al, Dreyband et al, Wang, and Wan similarly teaches a method *in which memory layout associated with the data is maintained as a flat layout* (See table Table B and D of Wan: whereas, Table B represents an XML instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes in a flat layout (index/list)).

With regards to claim 57, which depends on claim 32, for a system performing a method similar to the method of claim 26, is rejected under the same rationale.

With regards to claim 58, which depends on claim 32, for a system performing a method similar to the method of claim 28, is rejected under the same rationale.

With regards to claim 61, which depends on claim 33, for a computer program product performing a method that is similar to the method of claim 26, is rejected under the same rationale.

With regards to claim 62, which depends on claim 33, for a computer program product performing a method that is similar to the method of claim 28, is rejected under the same rationale.

7. Claims 5, 6, 7, 9, 22, 39, 41, 44, 46, 49, 53, 55, and 59 are rejected under 35 U.S.C. 103(a) as being unpatentable over Warshavsky et al (US Patent: 6,732,095 B1, issued: May 4, 2004, filed: Apr. 13, 2001) in view of Dreyband et al (US Patent Application: US 2001/0029604 A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), in view of Wang ("Discovering Typical Structures of Documents: A Road Map Approach", publisher: ACM, published: 1998), and further in view of JAXB (Sun Microsystems, pages 58, and 74, published: January 8, 2003).

With regards to claim 5, which depends on claim 1, Warshavsky et al does not explicitly teach a method *in which direct mapping is performed to an intended data type for the element.*

However, JAXB teaches a method *in which direct mapping is performed to an intended data type for an element* (page 74: whereas, based on the data type disclosed in a schema, a mapping is performed to an intended Java data type, as shown in the example, an 'int' datatype in the schema is mapped to an 'int' Java datatype, and a 'float' datatype in the schema is mapped to a 'float' Java datatype).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky's system to include a method of directly

mapping datatypes as taught by JAXB. The combination of Warshavsky, Dreyband et al, Wang et al, and JAXB would have allowed an application that utilized Warshavsky's system, to have accessed data more efficiently, without the overhead of any additional datatype mapping steps.

With regards to claim 6, which depends on claim 5, the combination of Warshavsky et al, Dreyband et al, Wang, and JAXB teaches *the conversion* (as similarly explained in the rejection for claim 5), and JAXB further teaches the conversion further includes a method in *which a conversion to a string datatype is not performed when mapping to the intended datatype* (page 74: whereas, an 'int' mapping is performed).

With regards to claim 7, which depends on claim 5, Warshavsky does not explicitly teach a method *in which the mapping is a close-matching datatype.*

However, JAXB teaches a method in *which the mapping is a close matching datatype* (page 58: whereas, mapping to a javatype is based on the defined schema datatype. As shown, an 'unsigned int' datatype in a schema is mapped to the closest matching datatype in Java, which is a 'long').

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky's XML access system to further provide a closest matching datatype that is supported for a particular system/application. The combination of Warshavsky et al, Dreyband et al, Wang et al, and JAXB would have

allowed Marcy's system operate flexibly on various different platforms using different datatypes.

With regards to claim 9, which depends on claim 1, the combination of Warshavsky et al, Dreyband et al, and Wang et al does not explicitly teach a method *in which the element is appropriate for association if it corresponds to a native datatype of the system in which the method is performed.*

Yet, JAXB teaches *in which the element is appropriate for association if it corresponds to a native datatype of the system in which the method is performed* (page 58, Table 5-1: whereas, all the schema datatypes disclosed are mapped to all supported Java datatypes. Furthermore, it is inherent that Java datatypes will be supported by the native datatypes available in the system through the use of the platform specific Java Virtual Machine).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky et al's system to further include a method for associating elements that map to Java datatypes for system independent datatype support. The combination of Warshavsky et al, Dreyband et al, Wang et al, and JAXB would have allowed Marcy's system to have operated independent of platform type.

With regards to claim 22, which depends on claim 18, the combination of Warshavsky et al, Dreyband et al, and Wang et al does not expressly teach *in which direct mapping is performed to an intended datatype for the* child node.

JAXB teaches a method *in which direct mapping is performed to an intended datatype for a child node/*element (page 74: whereas, based on the datatype disclosed in a schema, a mapping is performed to an intended Java datatype, as shown in the example, an 'int' datatype in the schema is mapped to an 'int' Java datatype, and a 'float' datatype in the schema is mapped to a 'float' Java datatype).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky et al's method of accessing child node data to further include a method of direct mapping to an intended datatype as taught by JAXB. The combination Warshavsky et al, Dreyband et al, Wang et al, and JAXB would have allowed Marcy's system to have performed faster child node data retrieval.

With regards to claim 39, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 5, is rejected under the same rationale.

With regards to claim 41, which depends on claim 30, for a system performing a method similar to the method performed by the method of claim 9, is rejected under the same rationale.

With regards to claim 44, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 5, is rejected under the same rationale.

With regards to claim 46, which depends on claim 31, for a computer program product performing a method similar to the method performed by the method of claim 9, is rejected under the same rationale.

With regards to claim 49, which depends on claim 34, for a system performing a method that is similar to the method of claim 22, is rejected under the same rationale.

With regards to claim 53, which depends on claim 35, for a computer program product performing a method that is similar to the method of claim 22, is rejected under the same rationale.

With regards to claim 55, which depends on claim 32, for a system performing a method similar to the method of claim 5, is rejected under the same rationale.

With regards to claim 59, which depends on claim 33, for a computer program product performing a method that is similar to the method of claim 5, is rejected under the same rationale.

8. Claims 11, 12, and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Warshavsky et al (US Patent: 6,732,095 B1, issued: May 4, 2004, filed: Apr. 13, 2001) in view of Dreyband et al (US Patent Application: US 2001/0029604

A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), in further view of Deutsch et al ("Storing Semistructured Data with STORED", pages 1-12, publisher: ACM, published: 1999), and further in view of Wang ("Discovering Typical Structures of Documents: A Road Map Approach", publisher: ACM, published: 1998).

With regards to claim 11, Warshavsky et al teaches a method comprising:

Storing the XML data in a database in a storage system, wherein one or more elements of a schema associated with the XML data are stored in one or more respective columns of the database (column 3, lines 58-62: whereas a DTD or schema is received, such that XML data is stored in one more columns);

Identifying an element associated with an instance of the XML data to access (column 4, lines 12-25: whereas, elements are stored into a database for later retrieval/access via a XML converter);

Determining, by using a processor, if the element has been associated with a access procedure corresponding to the element based at least in part upon a datatype of the element (Table 1: whereas, valid string datatypes are stored, while undefined string datatypes are ignored), wherein certain datatypes of elements within the database are designated as not eligible for the access procedure such that the element having at least one of the certain datatypes was not associated with the access procedure (Table 1: whereas, undefined string datatypes are ignored and not accessed through an access procedure);

If the element has been associated with the access procedure, then using the access procedure to access an instance of the XML data associated an instance of the XML data associated with the element such that the access procedure returns an appropriate datatype for the element without converting the datatype of the element (column 4, lines 12-20, Table 3: whereas, the appropriate string data type element values are returned/accessed using a XML converter), wherein direct access comprises accessing a column of the database for the element to access an instance of the XML data associated with the element *element* (column 3, lines 50-56, column 4, lines 13-17: whereas, a column of a database is referenced);

Storing the element in a volatile or non-volatile computer readable medium or displaying the element on a display device.

However, Warshavsky et al does not expressly teach the access procedure is a *named* access procedure, direct access … without progressive traversal of a hierarchy of elements defined in the schema , and also if the element has not been associated with the named access procedure*, then using a DOM API to access the instance of the XML data.*

Yet, Dreyband teaches a *named* access procedure (Figure 3, paragraph 0029: whereas, elements defined in the xml file are identified, such as the element used in Dreyband's example xml file labeled: "name");.

It would have been obvious to one of the ordinary skill in the art a the time of the invention to have modified Warshavsky et al's xml data access procedure, such that the accessing of node data can be accomplished through named access, as taught by

Dreyband et al. The combination of Warshavsky et al and Dreyband et al would have allowed Warshavksy et al to have implemented "the ability to effectively map a semantically descriptive language, such as Schema, into an object oriented language, without losing the relational characteristics of complex data structures (Dreyband et al, paragraph 0008).

However, the combination of Warshavsky et al and Dreyband et al do not expressly teach direct access … without progressive traversal of a hierarchy of elements defined in the schema , and … that if the element has not been associated with the named access procedure, *then using a DOM API to access the instance of the XML data.*

Yet, Deutsch et al teaches if the element has not been associated with the named access procedure, *then using a DOM API to access the instance of the XML data.*

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky et al and Dreyband et al's named access procedure, such that a DOM API can be used to access the instance of XML data, as taught by Deutsch et al. The combination would have allowed Warshavsky et al to have implemented a mapping that is always lossless, when semi-structured data do not fit the schema (Deutsch et al, page 1).

However, although Warshavsky et al supports compressing tables of mapping data (column 3, lines 64-67), the combination of Warshavsky et al, Dreyband et al, and Deutsch et al does not expressly teach *without progressive traversal of a hierarchy of elements defined in the schema*.

Yet, Wang teaches *without progressive traversal of a hierarchy of elements defined in the schema* (page 149, section 3, left column: whereas, the mapping of a hierarchy is compressed, such that frequently accessed nodes are mapped, rather than mapping all of the hierarchy of elements).

It would have been obvious to have modified Warshavsky et al, Dreyband et al, Deutsch et al's method for named access procedure, such that the named access is made possible without having to progressively traverse all the nodes in a hierarchy, as taught by Wang. The combination would have allowed Warshavsky et al to have "reduced the disk access" and access frequently accessed data more efficiently without linear traversal of the entire schema (Wang, page 149, section 3, column 1, and page 154, section 6, left column).

With regards to claim 12, which depends on claim 11, the combination of Warshavsky et al, Dreyband et al, and Wang et al teaches, *in which a schema for the XML data is known apriori and the named access procedure is based upon analysis of the schema,* as similarly explained in the rejection for claim 1, and is rejected under similar rationale.

With regards to claim 14, which depends on claim 11, the combination of Warshavsky et al, Dreyband et al, and Wang et al teaches a method *in which other elements of the data not presently needed are not loaded into memory* (as similarly explained in Wang et al for the rejection of claim 1; since infrequently used elements are opted to not be loaded into memory), and is rejected under similar rationale.

9. Claims 15, 16, and 17 are rejected under 35 U.S.C. 103(a) as being unpatentable over Warshavsky et al (US Patent: 6,732,095 B1, issued: May 4, 2004, filed: Apr. 13, 2001) in view of Dreyband et al (US Patent Application: US 2001/0029604 A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), in view of Deutsch et al ("Storing Semistructured Data with STORED", pages 1-12, publisher: ACM, published: 1999), in view of Wang ("Discovering Typical Structures of Documents: A Road Map Approach", publisher: ACM, published: 1998), and further in view of Wan (US Patent: 2003/0233618 A1, published: Dec. 18, 2003, filed: Jun. 16, 2003, Foreign priority: Jun. 17, 2002).

With regards to claim 15, which depends on claim 11, the combination of Warshavsky et al, Dreyband et al, Deutsch et al, and Wang do not teach a method in which *the element is at a known offset from a parent location*.

However, Wan teaches *the element is at a known offset from a parent location* (Table B and D: whereas, Table B represents an XML instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes using offsets based off of the memory location of a parent node).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky et al's application for accessing node data to further include the method for accessing child nodes via offsets based off of a location

of the corresponding parent nodes as taught by Wan. The combination of Warshavsky et al, Dreyband et al, Wang et al, Deutsch et al, and Wan would have allowed Warshavsky et al's system to have been able to have reduced iterative traversal of an XML instance.

With regards to claim 16, which depends on claim 15, the combination of Warshavsky et al, Dreyband et al, Deutch et al, Wang, and Wan similarly teach *in which the known offset is managed independently of the XML data* (as shown in table D of Wan), and is rejected under similar rationale.

With regards to claim 17, which depends on claim 11, the combination of Warshavsky et al, Dreyband et al, Deutsch et al, Wang, and Wan similarly teaches a method *in which a memory layout associated with the XML data is maintained as a flat layout* (See table Table B and D of Wan: whereas, Table B represents an XML instance (which includes child nodes), while Table D shows a method for accessing a memory location of the child nodes in a flat layout (index/list)).

10. Claim 13 is rejected under 35 U.S.C. 103(a) as being unpatentable over Warshavsky et al (US Patent: 6,732,095 B1, issued: May 4, 2004, filed: Apr. 13, 2001) in view of Dreyband et al (US Patent Application: US 2001/0029604 A1, published: Oct. 11, 2001, filed: Apr. 27, 2001), in view of Deutsch et al ("Storing Semistructured Data with STORED", pages 1-12, publisher: ACM, published: 1999), in view of Wang

("Discovering Typical Structures of Documents: A Road Map Approach", publisher: ACM, published: 1998), and further in view of JAXB (Sun Microsystems, pages 58, and 74, published: January 8, 2003).

With regards to claim 13, which depends on claim 11, Warshavsky et al does not explicitly teach a method *in which direct mapping is performed to an intended data type for the element.*

However, JAXB teaches a method *in which direct mapping is performed to an intended data type for an element* (page 74: whereas, based on the data type disclosed in a schema, a mapping is performed to an intended Java data type, as shown in the example, an 'int' datatype in the schema is mapped to an 'int' Java datatype, and a 'float' datatype in the schema is mapped to a 'float' Java datatype).

It would have been obvious to one of the ordinary skill in the art at the time of the invention to have modified Warshavsky's system to include a method of directly mapping datatypes as taught by JAXB. The combination of Warshavsky, Dreyband et al, Deutsch et al, Wang et al, and JAXB would have allowed an application that utilized Warshavsky's system, to have accessed data more efficiently, without the overhead of any additional datatype mapping steps.

Response to Arguments

11. Applicant's arguments with respect to claims 1-62 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

12. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to WILSON TSUI whose telephone number is (571)272-7596. The examiner can normally be reached on Monday - Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Stephen Hong can be reached on (571) 272-4124. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Stephen S. Hong/
Supervisory Patent Examiner, Art Unit 2178

/Wilson Tsui/
Patent Examiner
Art Unit: 2178
December 18, 2009